CHAPTER 05

EXPLAINING THE BOOT PROCESS

Boot and Startup Processes in Linux Systems

1. Introduction
2. The Firmware Startup
3. Linux bootloaders
4. System recovery options

1. Introduction

When you turn on the power to your Linux system, it triggers a series of events that eventually leads to the login prompt.

1.1 Following the Boot Process

1. Workstation firmware starts
    I. Performs a quick check of the hardware; POST
    II. Looks for a bootloader program to run from a bootable device
2. The bootloader runs and determines what Linux kernel program to load
3. The kernel program loads into memory
    - starts the necessary background programs required for the system to operate

1.2 Viewing the Boot Process

- Boot kernel messages are stored into
  - a buffer in memory, called the *kernel ring buffer*
    - could be reviewed using the dmesg command; *sudo dmesg*
  - a log file, usually in the /var/log folder
    - Sudo more /var/log/boot.log

2. The Firmware

- Control how to find hardware and how the OS starts
  - On older workstations, *Basic Input/Output System (**BIOS**)*.
  - On newer workstations, *Unified Extensible Firmware Interface (**UEFI**)*

2.1 The BIOS Startup

- A limitation: BIOS could read only one sector's from a hard drive into memory.

1. BIOS runs a *bootloader* program
    - produce a small menu allowing the user to boot between multiple OSs
    - allow you to load the bootloader program from several locations:
    - Internal hard drive, External hard drive, CD/DVD drive, or USB memory
    - Network server using either TFTP, NFS, HTTP, or FTP

2. Bootloader program, points to the location of the OS kernel file
    - has a configuration file
    - *located in the Master Boot Record (MBR)*

2.2 The UEFI Startup

- specifies a special disk partition, called the *EFI System Partition (ESP)*, to store bootloader programs
    - allows for any size of bootloader program, plus the ability to store multiple bootloader programs for multiple operating systems
- ESP is typically mounted in the /boot/efi directory, using the .efi extension
- boot manager, a built-in mini-bootloader, allows you to configure which bootloader program file to launch
    - efibootmgr

3. Linux Bootloaders

- Main bootloaders that have been used by default in Linux distributions:
  - **Linux Loader (LILO)**
    - doesn't work with UEFI systems, so it has limited use on modern systems
  - **Grand Unified Bootloader (GRUB) Legacy**
    - default bootloader for all Linux, whether it runs on BIOS or UEFI
  - **GRUB2, Supports advanced features**
    - ability to load hardware driver modules
    - using logic statements to dynamically alter the boot menu options

3.1 GRUB Legacy

- GRUB Legacy allows you to select multiple kernels and/or operating systems using
  - menu interface
    - provides options for each kernel or operating system you want to boot with
  - an interactive shell
    - provides a way for you to customize boot commands on the fly

3.1.1 Configuring GRUB Legacy

- GRUB Legacy system stores the menu commands in a text configuration file /boot/grub/grub.conf
  - GRUB Legacy configuration file consists of two sections:
    - Global definitions
    - Operating system boot definitions
  - ***Sample configuration file***

    default 0
    timeout 10
    color white/blue yellow/blue
    title Ubuntu Linux
    root (hd1,0)
    kernel (hd1,0)/boot/vmlinuz
    initrd /boot/initrd
    title Windows
    rootnoverify (hd0,0)

3.2 GRUB2

- GRUB2 system changes the configuration file name to grub.cfg and stores it in the /boot/grub/ folder
  - Allows to have both GRUB Legacy and GRUB2 installed at the same time
  - Some Red Hat-based Linux distributions make a symbolic link to this file in the /etc/grub2.cfg file; for easy reference

3.2.1 Configuring GRUB2

- Here's an example of a sample GRUB2 configuration file:

```
menuentry "Ubuntu Linux" {
set root=(hd1,1)
linux /boot/vmlinuz
initrd /initrd
}
menuentry "Windows" {
set root=(hd0,1)
}
```

- You should never modify that file. Instead, there are separate configuration files stored in the/etc/default
  - then sudo update-grub / sudo update-grub2
  - each boot option has individual configuration file
  - For global commands, use the /etc/default/grub configuration file

3.2.2 Installing GRUB2

- You don't need to install GRUB2; you simply rebuild the main installation file by running the grub2-mkconfig program

  ```
  sudo grub-mkconfig -o /boot/grub/grub.cfg
  ```

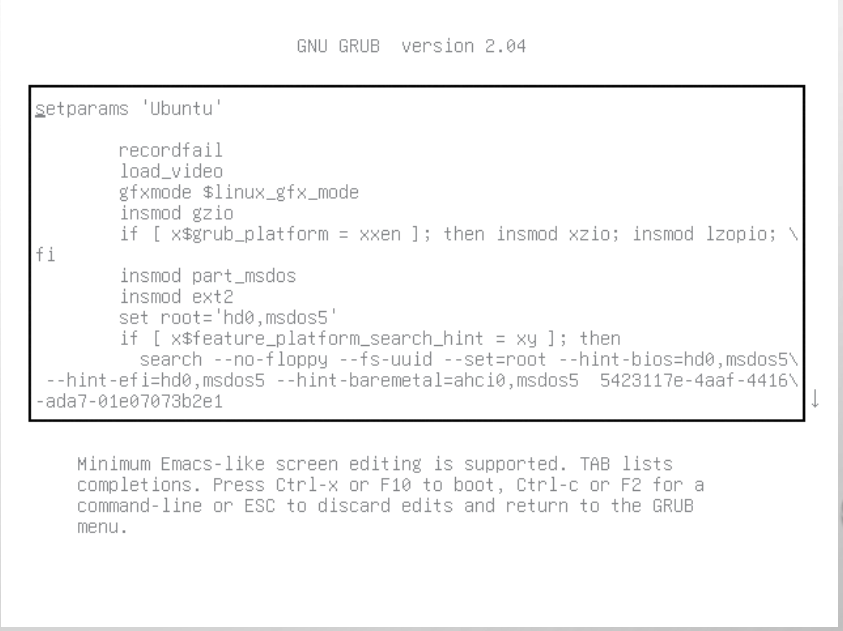
- sudo update-grub2, includes both:
  - grub2-mkconfig
  - grub2-install

3.2.3 Interacting with GRUB2

- shows the boot options defined in the configuration file.

- hold down the Shift key when the system first boots, that will display the GRUB boot menu

- can use arrow keys to switch between boot options, the E key to edit a boot entry, or the C key to bring up the GRUB2 command line to submit interactive boot commands.



3.3 Alternative Bootloaders

- *Syslinux project* includes five separate bootloaders that have special uses in Linux:
  - **SYSLINUX:** A bootloader for systems that use the Microsoft FAT filesystem
    - popular for booting from USB memory sticks
  - **EXTLINUX:** A mini-bootloader for booting from an ext2, ext3, ext4, or btrfs filesystem
  - **ISOLINUX:** A bootloader for booting from a LiveCD or LiveDVD
  - **PXELINUX:** A bootloader for booting from a network server; TFTP, NFS, HTTP, or FTP
  - **MEMDISK:** A utility to boot older DOS OSs from the other SYSLINUX bootloaders

4. System Recovery

- Plenty of things can go wrong in the Linux startup process, but most issues come down to two categories:
  - **Kernel failures (Kernel panic)**
    - Linux kernel stops running in memory, causing the system to crash
    - Can be fixed by using alternative boot method and editing the necessary files
  - **Drive failures**
    - may not be fatal, possible to recover from a corrupted root drive

4.1 Kernel Failures

- **Selecting Previous Kernels at Boot**
  - Most Linux distributions automatically keeping the most recent older kernel available in the boot menu when adding a new kernel
- **Single-User Mode; for the root user account**
  - by adding the single command to the Linux line in the boot menu commands
  - the system will boot into runlevel 1, using the Systemd startup method
- **Passing Kernel Parameters**
  - Specify different hardware settings as additional parameters to the kernel in the Linux command and then boot from that entry in the GRUB menu